CLAIMS:

1. A graphics processor configured to produce data for multiple output buffers, each output buffer associated with a unique output buffer identifier, comprising:

a fragment processing pipeline configured to process graphics data to produce processed graphics data for the multiple output buffers and determine at least one output buffer identifier associated with the processed graphics data;

a shader read interface configured to read processed graphics data associated with an output buffer identifier from an output buffer stored in a memory; and

a write interface configured to write processed graphics data associated with at least one output buffer identifier to an output buffer stored in the memory.

2. The graphics processor of claim 1, further comprising a geometry processor configured to process graphics data.

3. The graphics processor of claim 1, wherein the at least one output buffer identifier is determined by a fragment program executed in the fragment processing pipeline.

4. The graphics processor of claim 1, wherein an output buffer includes data represented in two or more data formats.

5. The graphics processor of claim 1, wherein the output buffer identifier is readable and writable by the fragment processing pipeline.

6. The graphics processor of claim 1, wherein any of the multiple output buffers is selected for display.

7. The graphics processor of claim 1, wherein the fragment processing pipeline includes multiple registers, each register capable of outputting data to one or more of the multiple output buffers.

8. The graphics processor of claim 1, wherein the fragment processing pipeline determines an address associated with the processed fragment data, the address corresponding to a specific location in an output buffer.

9. The graphics processor of claim 1, wherein the graphics processor resides within a computing system including a host processor.

10. A method of processing fragment data for multiple output buffers, comprising:

processing fragment data as specified by a fragment program to produce processed fragment data for the multiple output buffers;

determining an output buffer identifier associated with the processed fragment data; and

storing the processed fragment data in an output buffer corresponding to the output buffer identifier.

11. The method of claim 10, further comprising:

reading the processed fragment data from the output buffer using the output buffer identifier.

12. The method of claim 10, wherein the processed fragment data stored in the output buffer includes fragment depth data.

13. The method of claim 10, wherein the processed fragment data stored in the output buffer includes an index.

14. The method of claim 13, wherein the index is a shader identifier.

15. The method of claim 13, wherein the index is a pointer to a fragment program.

16. The method of claim 10, wherein the processed fragment data stored in the output buffer includes displaced mesh data.

17. A system for processing graphics data using deferred shading, comprising:

means for processing geometry to produce visible fragment data for visible fragments;

means for storing depth values of the visible fragments in an output buffer;

means for storing a portion of the visible fragment data in one or more additional output buffers; and means for shading the visible fragments using the portion of the visible fragment data read from the one or more additional output buffers.

18. The system of claim 17, wherein the portion of the fragment data includes an index.

19. The system of claim 17, wherein the portion of the fragment data includes lighting parameters.

20. The system of claim 17, wherein the portion of the fragment data includes color data.